REMARKS

The Examiner is thanked for the performance of a thorough search.

Prior to this response, Claims 1-38 were pending in the application. By this amendment, Claims 1, 14, 19, 20, 32 and 38 are amended, and no claims are added or cancelled. Hence, Claims 1-38 are currently pending in the application.

SUMMARY OF THE REJECTIONS/OBJECTIONS

Claims 1-7, 10-16, 18-26, 29-34, and 36-38 were rejected under 35 U.S.C. § 102(e) as allegedly anticipated by Nock et al. ("*Nock*", U.S. Patent No. 6,513,115).

Claims 8, 9, 27 and 28 were rejected under 35 U.S.C. § 103(a) as allegedly unpatentable over *Nock* in view of Prasad et al. ("*Prasad*", U.S. Patent No. 6,539,381).

Claims 17 and 35 were rejected under 35 U.S.C. § 103(a) as allegedly unpatentable over *Nock* in view of Croft et al. ("*Croft*", U.S. Patent No. 6,470,436).

THE REJECTIONS BASED ON THE PRIOR ART

Rejection under 35 U.S.C. 102(e)

Claims 1-7, 10-16, 18-26, 29-34, and 36-38 were rejected under 35 U.S.C. § 102(e) as allegedly anticipated by *Nock*. This rejection is traversed, on the basis that a *prima facie* case of anticipation was not established.

Nock fails to teach or suggest every limitation of Claim 1 and, therefore, does not fairly anticipate Claim 1. As indicated in the Office Action, *Nock* discloses configuring an existing server process without the need to end and restart the server process (Abstract). *Nock* defines a server process as "a software application or routine that is started on a well-known computer that, in turn, operates continuously, waiting to connect and service the requests from various clients" (col. 1, lines 30-36). Claim 1 of the application recites a computer-implemented

method for reconfiguring an application executing on a computer system without restarting the computer system. Thus, Applicants concede that, at that level of detail, *Nock* sounds similar to the embodiment recited in Claim 1. However, there is at least one fundamental difference between the teachings of *Nock* and Claim 1, described hereafter.

Nock discloses that a new server process is first installed with new configuration information (col. 4, lines 51-56; block 210 of Fig. 2). Then, it is determined whether or not an existing server process is running by determining whether the existing server process is listening to a particular port. If an existing server process is bound to the port, then the existing server process is executing (col. 4, line 64 through col. 5, line 26; block 220 of Fig. 2). If there is no existing server process running, then the new server process becomes the current server process configured with the new configuration information and begins accepting client connections (col. 5, lines 39-43; block 230 of Fig. 2). If there is an existing server process running, the new server process assumes the role of a client by issuing a configuration request, which should lead to updating the old configuration information of the existing server process with the new configuration information (col. 5, lines 44-67; block 240 of Fig. 2).

Significantly, *Nock* states that the "method 200 avoids a conflict between two server processes operating simultaneously with different configuration parameters" (col. 5, lines 65-67). *Nock* does not describe *how* such conflicts are avoided. However, the only reasonable interpretation based on what *Nock* does describe is that conflicts between two server processes operating simultaneously with different configuration parameters are avoided because *Nock* simply does not allow two differently-configured server processes to run simultaneously. This limitation is clear based on the description of Fig. 2, as summarized above. For example, the logic of blocks 220, 230 and 240 shows that (a) if no existing server process is running, then the new server process simply begins to accept new client connection; and (b) if an existing server

process is running, then a configuration update process is initiated for the existing server process. This limitation is further exemplified by block 250, which shows that all client requests are processed by the existing server process, i.e., there is only one server process that is capable of servicing client requests at any given time.

By contrast, Claim 1 recites that the second application configuration is provided for servicing new requests related to the application while maintaining the first application configuration for servicing, concurrently with new requests, existing requests related to the application. Hence, multiple differently-configured application "instances" can be running simultaneously at any given time. That is, the "old" application services existing requests while, concurrently, the "current" application services new requests. For example, the old application services requests from connections/sessions that started before the application reconfiguration process, and the new application concurrently services requests from new connections/sessions that were initialized after the reconfiguration process commenced (see, e.g., paragraph [0017], [0035], [0065], [0076], [0084] and [0085]).

This ability to concurrently service requests by two different versions of an application is facilitated in part through use of a global variable, such as the current configuration pointer 112 of FIG. 1A. In providing the service, the server consults the global variable to obtain the pointer to the configuration data structures, and associates that pointer with the server connection, and thus the user session (paragraph [0014]). Further, while the server is constructing the new set of data structures, the global variable continues to store a pointer to the first set of data structures. Hence, all connections established by the server while the new data structures are being constructed are associated with the first set of data structures. Consequently, all requests, events, and activities received on those connections, including those

related to the application, are processed by the server in accordance with the first set of data structures (paragraph [0016]).

The foregoing remarks show that the manner in which *Nock* reconfigures a server process without ending and restarting the server process is clearly different than the embodiment recited in Claim 1 for reconfiguring an application without restarting the computer system (i.e., the "server") on which the application is executing. *Nock* does not teach or suggest an implementation that allows for multiple differently-configured versions of the same application to run on the same machine at the same time, both concurrently servicing requests. In fact, the primary premise or objective of *Nock* appears to be avoiding having two differently-configured versions of the same application running on the same machine at the same time. *Nock* appears silent as to *how* the configuration is changed, at block 280 of Fig. 2, without causing any interruption in service. The embodiment of Claim 1 provides for a system with uninterrupted service through use of more than one "live" (i.e., currently providing service) application instance running on the system at the same time, with requests being serviced accordingly.

For the foregoing reasons, *Nock* does not anticipate Claim 1 and therefore Claim 1 is patentable thereover. Withdrawal of the rejection of Claim 1 is respectfully requested.

Independent Claim 14 recites a first request that is associated, via a global variable, with a first application configuration and a second request that is associated, via a global variable, with a second application configuration. See, e.g., paragraphs [0013], [0014], [0016] and [0017]. *Nock* does not teach or suggest use of a global variable for storing a reference or pointer to a configuration data structure that controls all requests, events, and activities received on those connections that are associated with a particular reference or pointer. Further, *Nock*

does not teach or suggest use of such a global variable to enable concurrent servicing of requests by two or more differently-configured versions of the same application, by associating different sessions/connections with different versions of the application. Hence, *Nock* does not anticipate Claim 14 and therefore Claim 14 is patentable thereover. Withdrawal of the rejection of Claim 14 is requested.

Independent Claim 19 recites that, while completing processing the one or more pending requests based on a former configuration, a new request is received, current configuration information accessed, and the new request processed accordingly. Thus, it is clear that requests can be simultaneously serviced according to both a former configuration and a new configuration. As discussed, *Nock* does not teach or suggest the forgoing and, therefore, *Nock* does not anticipate Claim 19. Withdrawal of the rejection of Claim 19 is requested.

Independent Claims 20 and 38 recite similar limitations as Claim 1. Hence, Claims 20 and 38 are patentable over the references of record for at least the same reasons as Claim 1. Withdrawal of the rejection of Claims 20 and 38 is requested.

Independent Claim 32 recites similar limitations as Claim 14. Hence, Claim 32 is patentable over the references of record for at least the same reasons as Claim 14. Withdrawal of the rejection of Claim 32 is requested.

(A) Dependent Claims 2-7 and 10-13 depend from Claim 1; (B) dependent Claims 15, 16 and 18 depend from Claim 14; and (C) dependent Claims 21-26, 29-34, 36 and 37 depend from Claim 20. Thus, these claims are patentable over the references of record for at least the

same reasons as the claims from which they respectively depend. Withdrawal of the rejection of the foregoing claims is requested.

Rejection under 35 U.S.C. 103(a)

Claims 8, 9, 27 and 28 were rejected under 35 U.S.C. § 103(a) as allegedly unpatentable over *Nock* in view of *Prasad*; and Claims 17 and 35 were rejected under 35 U.S.C. § 103(a) as allegedly unpatentable over *Nock* in view of *Croft*. These rejections are traversed, on the basis that a *prima facie* case of obviousness was not established.

(A) Dependent Claims 8 and 9 depend from Claim 1; (B) dependent Claim 17 depends from Claim 14; and (C) Claims 27, 28 and 35 depend from Claim 20. As with the anticipation rejection, the Office Action primarily relies on the disclosure of *Nock* to substantiate the obviousness rejections. However, it is already shown above that *Nock* does not disclose all the limitations for which the Action is relying on *Nock*. Thus, these claims are patentable over the references of record for at least the same reasons as the claims from which they respectively depend, that is, due to the absence of relevant teachings from *Nock*. Furthermore, neither *Prasad* nor *Croft* cure these deficiencies in *Nock*. Therefore, withdrawal of the respective rejections of Claims 8, 9, 17, 27, 28 and 35 is requested.

CONCLUSION

For the reasons set forth above, it is respectfully submitted that all of the pending claims (1-38) are now in condition for allowance. Therefore, the issuance of a formal Notice of Allowance is believed next in order, and that action is most earnestly solicited.

The Examiner is respectfully requested to contact the undersigned by telephone if it is believed that such contact would further the examination of the present application.

Please charge any shortages or credit any overages to Deposit Account No. 50-1302.

Respectfully submitted,
HICKMAN PALERMO TRUONG & BECKER LLP

Date: 4/12/05

John D. Henkhaus
Reg. No. 42,656

2055 Gateway Place, Suite 550
San Jose, CA 95110-1089
(408) 414-1080
Facsimile: (408) 414-1076

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Mail Stop Amendment, Commissioner for Patents, P. O. Box 1450, Alexandria, VA 22313-1450

on 4/12/05 by Darci Sakamoto